Ε. Μ. Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ.
& Μηχ. Υπολογιστών.
Ε. Ζάχος, Ν. Παπασπύρου,
Β, Καντερέ, Π. Ποτίκας

ΕΠΩΝΥΜΟ:	1	
ΟΝΟΜΑ:	2	
ΑΡ. ΜΗΤΡΩΟΥ:	3	
ΕΞΑΜΗΝΟ:	4	
ΟΜΑΔΑ ΕΡΓ:	5	
ΑΜΦΙΘΕΑΤΡΟ:	6	
ΘΕΣΗ:	ΣΥΝΟΛΟ	

B

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Η/Υ

Κανονική εξέταση, Φεβρουάριος 2018

Κανονισμός εξέτασης: *1) Υποχρεούστε να δείξετε στον επιτηρητή όταν σας ζητηθεί τη φοιτητική σας ταυτότητα ή άλλο αποδεικτικό της ταυτότητάς σας με φωτογραφία. 2) Η εξέταση γίνεται με κλειστά βιβλία και σημειώσεις. 3) Δεν μπορείτε να χρησιμοποιείτε ηλεκτρονικές συσκευές. Αν έχετε μαζί σας κινητό τηλέφωνο, απενεργοποιήστε το και κρύψτε το.*

1. (8)

Να δείξετε σε **πίνακα** όλες τις **ενδιάμεσες τιμές** καθώς και τις **τιμές που τυπώνονται** από το παρακάτω πρόγραμμα C++ (εκτέλεση με το χέρι). Δεξιά, το ίδιο πρόγραμμα σε απλή C.

```
#include "pzhelp"

int a = 7, b = 1, c = 2;

PROC p(int a, int &b) {
  int c = 2*a;
  a = b-1;
  WRITELN(a, b, c);
  if (a <= c) {
    p(b+1, c);
    WRITELN(a, b, c);
  }
}

PROGRAM {
  p(c+1, a);
  WRITELN(a, b, c);
}
```

```
#include <stdio.h>

int a = 7, b = 1, c = 2;

void p(int a, int *b) {
  int c = 2*a;
  a = *b-1;
  printf("%d %d %d\n", a, *b, c);
  if (a <= c) {
    p(*b+1, &c);
    printf("%d %d %d\n", a, *b, c);
  }
}

int main() {
  p(c+1, &a);
  printf("%d %d %d\n", a, b, c);
  return 0;
}
```

2. (8)

Κατασκευάστε **συντακτικό διάγραμμα** που περιγράφει τις συμβολοσειρές της μορφής $a^i\, b^j\, c^{i+j}$, όπου $i, j \geq 0$. Μερικές τέτοιες συμβολοσειρές είναι οι εξής:

abcc *aabccc* *abbccc* *aaabbccccc* *aacc* *bbbccc*

Δηλαδή, το συντακτικό διάγραμμα πρέπει να περιγράφει τις συμβολοσειρές που:

- αρχίζουν με $i \geq 0$ φορές το γράμμα a
- συνεχίζουν με $j \geq 0$ φορές το γράμμα b και
- τελειώνουν με $i+j$ φορές το γράμμα c.

3. (10)

Απαντήστε στις παρακάτω ερωτήσεις πολλαπλής επιλογής μαυρίζοντας σε κάθε μία το πολύ ένα από τα τέσσερα κουτάκια. Κάθε σωστή απάντηση παίρνει 1,5 μονάδα. Κάθε λάθος απάντηση χάνει 0,5 μονάδα (αρνητική βαθμολογία). Κενές ή άκυρες απαντήσεις δεν προσθέτουν ούτε αφαιρούν μονάδες.

(α) Σε ένα ισοζυγισμένο δυαδικό δέντρο αναζήτησης με n στοιχεία, η αναζήτηση ενός στοιχείου:

☐ απαιτεί χρόνο $O(1)$ στη χειρότερη περίπτωση.
☐ απαιτεί χρόνο $\Omega(n)$ στη χειρότερη περίπτωση.
☐ απαιτεί χρόνο $O(1)$ στην καλύτερη περίπτωση.
☐ απαιτεί χρόνο $\Omega(n \log n)$ στη χειρότερη περίπτωση.

(β) Έστω ότι έχετε τρεις διαφορετικούς αλγορίθμους Α, Β και Γ, που επιλύουν το ίδιο πρόβλημα. Η πολυπλοκότητα του Α είναι $O(n^3)$, του Β είναι $O(n^2 \log^5 n)$, και του Γ είναι $O(n!)$. Ποιον από τους τρεις θα προτιμούσατε; (Θεωρήστε ότι μας ενδιαφέρουν μεγάλες τιμές του n.)

☐ τον Α
☐ τον Β
☐ τον Γ
☐ οποιονδήποτε από τους Α ή Β, δεν έχουν διαφορά

(γ) Ποια είναι η τιμή της μεταβλητής **t** στο τέλος της εκτέλεσης του ακόλουθου τμήματος προγράμματος;

```
int n = 1024, t = 0;
for (int i = 1; i <= n; i *= 2) {
  for (int j = 1; j <= i; j++) t++;
  n--;
}
```

☐ 1023 ☐ 1024 ☐ 2047 ☐ κανένα από τα προηγούμενα

(δ) Ποιο από τα παρακάτω προγράμματα τυπώνει **42**;

```
int k=3;
PROC proc1(int &n) {
  k *= n-1; WRITELN((n+1)*k);
}
PROGRAM { proc1(k); }
```

```
int k=3;
PROC proc2(int n) {
  k *= n-1; WRITELN((n+1)*k);
}
PROGRAM { proc2(k); }
```

☐ το αριστερό ☐ το δεξιό ☐ και τα δύο ☐ κανένα από τα δύο

(ε) Τι τυπώνει το παρακάτω πρόγραμμα;

```
PROGRAM {
  int *p = new int; *q = new int; *t = new int;
  *p=3; *q=5; *t=7;
  p=q; q=t;
  *t=*p**q ; *q*=*t;
  WRITELN(*t);
}
```

☐ 6 ☐ 8 ☐ 42 ☐ κανένα από τα προηγούμενα

(ζ) Ποια λογική πρόταση είναι σωστή αναλλοίωτη για το σημείο «1» του παρακάτω βρόχου;

```
int L = 0, R = 1000, x = 4217;
while (R-L > 1) { /* 1 */
  int M = (L+R)/ 2, z = M*M*M;
  if (z <= x) L = M; else R = M;
}
```

☐ $L < R-1$ και $z = (L+R)^3/8$
☐ $L < R-1$ και $L \le \sqrt[3]{x}$ και $R > \sqrt[3]{x}$
☐ $L < R-1$ και $L < \sqrt[3]{x}$ και $R \ge \sqrt[3]{x}$
☐ καμία από τις παραπάνω

(η) Ποια είναι η τιμή της μεταβλητής **n** στο τέλος της εκτέλεσης του ακόλουθου τμήματος προγράμματος;

```
int n = 999, p = 1;
do { n /= 2; p *= 2; } while (n > 4);
n += p;
```

☐ 131 ☐ 259 ☐ 1024 ☐ κανένα από τα προηγούμενα

ΠΡΟΧΕΙΡΟ

4. (10)

Αν γράψετε μόνο τη λέξη «ΚΕΝΟ» αντί λύσης σε αυτό το θέμα, θα πάρετε 2 μονάδες.

Θεωρούμε έναν μονοδιάστατο πίνακα **A** με **N** ακέραιους αριθμούς ($1 \le \mathbf{N} \le 1.000.000$), οι τιμές των οποίων είναι στο διάστημα από 0 έως **K**−1 ($1 \le \mathbf{K} \le \mathbf{N}$). Μας ενδιαφέρουν τα τμήματα του πίνακα **A** (δηλαδή διαδοχικοί όροι A[i], A[i+1] … A[i+k]) στα οποία εμφανίζονται **όλες οι τιμές** από 0 έως **K**−1. Ποιο είναι το μήκος του μικρότερου τέτοιου τμήματος;

Να γράψετε μία *κομψή και αποδοτική* συνάρτηση που δέχεται ως παραμέτρους τα **A**, **N** και **K**, και υπολογίζει το **ελάχιστο** μήκος ενός τμήματος του πίνακα **A** που να περιέχει όλες τις τιμές από 0 έως **K**−1. Αν δεν υπάρχει τέτοιο τμήμα, η συνάρτηση πρέπει να επιστρέφει 0.

Παράδειγμα 1: (**N** = 10, **K** = 3)

A = [0, 2, 0, 2, 0, 2, 2, 1, 1, 0]
allnums(**N**, **K**, **A**) = 4

Το τμήμα [0, 2, 2, 1], που είναι παραπάνω υπογραμμισμένο για διευκόλυνσή σας, έχει μήκος 4 και περιέχει όλους τους αριθμούς από 0 έως 2. Είναι το ελάχιστο τέτοιο τμήμα.

Παράδειγμα 2: (**N** = 10, **K** = 5)

A = [0, 1, 1, 4, 0, 2, 0, 1, 0, 2]
allnums(**N**, **K**, **A**) = 0

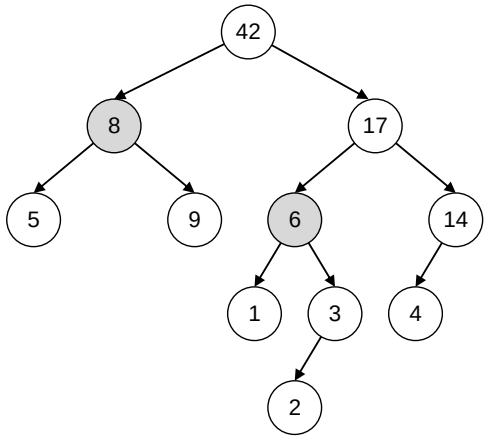
Στον παραπάνω πίνακα δεν υπάρχει τμήμα που να περιέχει όλους τους αριθμούς από 0 έως 4, γιατί η τιμή 3 δεν εμφανίζεται στον πίνακα.

Ερώτηση bonus (2 επιπλέον μονάδες): Ποια είναι η πολυπλοκότητα της λύσης σας; Εξηγήστε.

ΠΡΟΧΕΙΡΟ

5. (10)

Αν γράψετε μόνο τη λέξη «ΚΕΝΟ» αντί λύσης σε αυτό το θέμα, θα πάρετε 2 μονάδες.



Ορίστε τον τύπο **tree** του δυαδικού δέντρου που περιέχει στους κόμβους του ακέραιους αριθμούς, καθώς και τον τύπο **node** του κόμβου του.

Το ***ύψος*** (height) ενός κόμβου του δέντρου ορίζεται ως το μήκος του μακρύτερου μονοπατιού που ξεκινάει από τον κόμβο και καταλήγει σε φύλλο.

Το ***βάθος*** (depth) ενός κόμβου του δέντρου ορίζεται ως το μήκος του μονοπατιού που ξεκινάει από τον κόμβο και καταλήγει στη ρίζα του δέντρου.

Γράψτε μια *κομψή και αποδοτική* συνάρτηση που δέχεται ως παράμετρο ένα δέντρο και επιστρέφει το πλήθος των κόμβων που έχουν ***ίσο ύψος και βάθος***. Η συνάρτηση θα πρέπει να έχει ως επικεφαλίδα:

```
FUNC int count(node *t);
```

Για το παράδειγμα του διπλανού σχήματος, η συνάρτησή σας θα πρέπει να επιστρέφει 2 (οι δύο κόμβοι που είναι χρωματισμένοι με γκρίζο είναι αυτοί που έχουν ίσο ύψος και βάθος).

ΠΡΟΧΕΙΡΟ

6. (10)

Αν γράψετε μόνο τη λέξη «ΚΕΝΟ» αντί λύσης σε αυτό το θέμα, θα πάρετε 2 μονάδες.

Ζητείται ένα *κομψό και αποδοτικό* πρόγραμμα που διαβάζει από το αρχείο με όνομα "`file.txt`" ένα (μη κενό) κείμενο αποτελούμενο από πεζά γράμματα του λατινικού αλφαβήτου, κενά διαστήματα και αλλαγές γραμμής. Το πρόγραμμά σας πρέπει να εκτυπώνει στην οθόνη το κείμενο που προκύπτει από το αρχικό, αν κάθε λέξη αντικατασταθεί από το πλήθος των γραμμάτων της.

Παράδειγμα:

(κείμενο):

```
the first electronic computers were monstrous contraptions
filling several rooms consuming as much electricity as a
good size factory and costing millions of dollars
but with the computing power of a modern hand held calculator
```

(οθόνη):

```
3 5 10 9 4 9 12
7 7 5 9 2 4 11 2 1
4 4 7 3 7 8 2 7
3 4 3 9 5 2 1 6 4 4 10
```

ΠΡΟΧΕΙΡΟ